An Evaluation of Current Approaches for Web Service Composition

Sayed Gholam Hassan Tabatabaei Wan Mohd Nasir Wan Kadir Suhaimi Ibrahim
Department of Software Engineering, Faculty of Computer Science and Information Systems, University of Technology Malaysia (UTM), 81310 Skudai, Johor, Malaysia
gtsayed2@siswa.utm.my wnasir@utm.my suhaimiibrahim@utm.my

Abstract

Since many organizations recently decide to implement and publish their applications over Internet, the number of Web services has dramatically increased. In many cases, a single service is not sufficient to respond to the user's request. In order to tackle this problem, services have to be combined together. Therefore, composition of Web services is one of the recent critical issues. Several approaches have been presented, to tackle this problem. In this paper, we classify these approaches into four categories namely Workflow-based, AI-planning based, Syntactic-based, and Ontology-based. Then, we describe and compare these approaches using some criteria (like QoS, scalability, and correctness). The overall results indicate that some AI-planning and Ontology based approaches like HTN-DL and WSMO satisfy most of the criteria.

1. Introduction

The term "Web services" has been used very often nowadays. According to W3C, "A *Web service is a software system identified by a URI* [1]*, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols* "[2]. Another definition of Web service is provided by IBM [3], A Web service is a software interface that describes a collection of operations that can be accessed over the network through standardized XML messaging. It uses protocols based on the XML language to describe an operation to execute or data to exchange with another Web service.

Basically, Web service operation can be described as follows. First of all, a *client* program via a yellow page (UDDI) [3] finds a *Web services server* that can fulfill certain requirements, and acquire a detailed specification from WSDL [4] about the service. Then, the client sends a *request* to the server through a standard message protocol (SOAP) [5], and in return receives a *response* from the server. With interpreting XML tags, applications can interpret the operations and data much easier than conventional programming interface.

Nowadays, an increasing amount of companies and organizations implement their applications over Internet. For example, if a user wants to participate on one international conference, it is not sufficient to register, but he should also take care of booking a flight, reserving a hotel, renting a car, and so on. Thus, the ability to efficiently and effectively select and integrate inter-organizational and heterogeneous services on the Web at runtime is an important step towards the development of the Web service applications. Recent research studies how to specify them (in a formal and expressive enough language), how to (automatically) compose them, how to discover them (on the Internet) and how to ensure their correctness. We focus on Web Service composition (WSC).

When no atomic Web service (WS) can satisfy the user's requirements, there should be a possibility to combine existing services together in order to accomplish the request. This trend has inaugurated a considerable number of research efforts on the WSC both in academia and in industry.

A composite service, in many ways, is similar to a workflow [6]. The definition of a composite service includes a set of atomic services together with the control and data flow among the services. Similarly, a workflow has to specify the flow of work items. The dynamic workflow approaches provide the means to bind the abstract nodes with the concrete resources or services automatically. Some approaches based on AI planning, consider WS as a software component that

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

takes the input data (preconditions) and produces the output data (effects). Since the WS alters the state of the world after execution, the world state prerequisite for the service execution is the precondition, and the new state generated after the execution is the effect [10].

In the research related to Web services, several initiatives have been conducted with the intention to provide platforms and languages for WSC such as Business Process Execution Language for Web Services (BPEL4WS) [7]. Nowadays some languages have ability to support semantic representations of the WSs available on the Internet such as the Web Ontology Language for Web Services OWL-S [8] and the Web Service Modeling Ontology WSMO [9].Although all of these efforts, the WSC still is a highly complex task.

In this paper, we focus on the WSC problem and offer a survey of recent approaches that provide automation to Web service composition. The automation means that either the approach can generate the process model automatically, or the method can locate the correct services if an abstract process model is given [11]. We then compare them with respect to the set of criteria. By offering this overview and classification of existing proposals for Web service composition, as well as a constructive review of them, we hope to help service-composition designers and developers focus their efforts and deliver more usable solutions, while also addressing the technology's critical requirements.

2. Classification of the WSC approaches

We can classify the WSC approaches using the following four aspects:

2.1. Workflow-based WSC approaches

Workflow-based composition methods can be distinguished to the static and dynamic workflow generation [11]. The *Static Composition* means that the requester before starting the composition planning should build an abstract process model. The abstract process model includes a set of tasks and their data dependency. Each task contains a query clause that is used to search the single WS to fulfill the task. Thus, just the selection and binding of single WS is done automatically by software. However, in *Dynamic Composition*, creating process model and selecting single WSs are done automatically. The requester has to specify several constraints, such as the user's preference. In this section we describe two principal approaches, namely:

- *EFlow* [12] is a platform for the specification, enactment and management of WSC which uses a static workflow generation method. In that case, WSC is modeled by a graph that defines the order of execution among the nodes in the process. The graph is created manually but it can be updated dynamically. The graph may include service (represent the invocation of WS), decision (specify the alternatives and rules controlling the execution flow) and event nodes (enable service processes to send and receive several types of events). Arcs in the graph denote the execution dependency among the nodes. The definition of a service node contains a search recipe that can be used to query actual service. As the service node is started, the search recipe is executed, returning a reference to a specific service.
- *Polymorphic Process Model (PPM)* [13] uses a method that synthesizes the static and dynamic WSC. The static setting is supported by reference process-based multi-enterprise processes. These processes include abstract sub processes that have functionality description but lack implementation. The abstract subprocesses are implemented by service and bined at runtime. The dynamic part of PPM is supported by service-based processes. Here, a service is modeled by a state machine that specifies that possible states of a service and their transitions. Transitions are caused by service operation invocations or internal service transitions. In the setting, the dynamic service composition is enabled by the reasoning based on state machine.

2.2. AI-Planning-based WSC approaches

Currently, several approaches based on AI planning have been presented to solve the problem of WSC. Most of these approaches rely on the model of state-transition systems. In this system there are finite or recursively countable set of states, actions and events along with a transition function that maps a state, action, event tuple to a set of states. The goal of planning is to find which actions to apply to which states in order to achieve some objective, starting from some given situation.

Basically, classical planning is based on the initial modeling of the STRIPS [30] system. In this representation a state is represented by a set of ground literals expressed in a first-order language. An action is an expression specifying which first-order literals belong to the state in order for the action to be applicable, and which literals the action will add or

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

remove in order to make a new world state. A planning operator is a triple o = (N, P, E), where N, name of the operator, P is the precondition of the operator expressed as a conjunction of set of literals and E is the effects of the operators which can be positive or negative. An operator o is applicable in a state s when the preconditions are satisfied in the state. Applying the effects of an operator is done by adding or deleting entries from the database.

In the terms of WSs, since services have preconditions and effects that are expressed as logical conditions, several WS languages describe services in ways which are influenced by AI planning. Using this similarity, it is possible to deal with WSs as planning operators and use a causal planner to generate WSC. Therefore, each WS is first translated to a planning operator, the objective is expressed as a logical condition, and the planner generates a plan which is essentially a sequence of WS instances.

In the following we introduce some of well-known WSC approaches based on AI planning. Excellent surveys of AI-planning-based approaches to tackle the problem of WSC can be found in [11, 31].

- *Situation Calculus* is a first-order language for reasoning about action and change. In the situation calculus, the state of the world is described by functions and relations (fluents) relativized to a situation s, such as $f(x,s)$. The function $do(a,s)$ maps a situation s and an action a into a new situation. A situation is simply a history of the primitive actions performed from an initial, distinguished situation S_0. Golog is a high-level logic programming language based on the situation calculus that enables the representation of complex actions. It builds on top of the situation calculus by providing a set of extra-logical constructs for assembling *PrimitiveActions*, defined in the situation calculus, into *ComplexActions* that are compositions of individual actions.

McIlraith et. al. [32] adapt and extend the Golog language for automatic construction of WSs. Actually, this approach is based on the notion of generic procedures. The authors address the WSC problem through the provision of high-level generic procedures and customizing constraints. Golog is adopted as a natural formalism for representing and reasoning about this problem. The general idea of this method is that software agents could reason about Web services to perform automatic Web service discovery, execution, composition and inter-operation. The authors conceive each Web service as an action. *PrimitiveActions* are conceived as either world-altering actions that change the state of the world or information-gathering actions that change the agent's state of knowledge. The agent knowledge base provides a logical encoding of the preconditions and effects of the WS actions in the language of the situation calculus. A composite service is a set of single services which connected by procedural programming language constructs (if-then-else, while and so forth). The composition system uses an augmented Golog interpreter that combines online execution of sensing actions with offline simulation of world altering services.

One advantage of using situation calculus is the additional expressivity and the ability to do arbitrary reasoning about first-order theories. However, Golog implementation uses regression to reason about actions, i.e. to solve executability and projection problems. According to [33], translating OWL-S descriptions to situation calculus and applying regression yields a standard first-order theory which is not in the scope of what Golog can handle without calling a general first-order theorem prover. Furthermore, the programs, which are enabled by Golog and defined as macros, are complied away. So it is impossible to describe non-functional attributes of such programs or use these attributes for flexible matching.

- *HTN-DL*: Sirin [34] proposes the HTN-DL formalism which combines Hierarchical Task Network (HTN) planning, and Description Logics (DL) to automatically overcome the problem of WSC which are described with Web Ontology Language (OWL).

The hierarchical structure of HTN planning domains can describe composite service descriptions. Composite Web Services can be mapped to HTN methods whereas atomic WSs are mapped to HTN operators. HTN-style domains fit in well with the loosely coupled nature of WSs: different decompositions of a task are independent so the designer of a method does not have to have close knowledge of how the further decompositions will go or how prior decompositions occurred. The DL is used to describe both actions and states with an expressive knowledge representation language. The service categorization and non-functional attributes of services are described in a task ontology that allows flexible matchmaking. The state of the world is also represented as a DL knowledge base. HTN-DL also differentiates between world-altering effects and knowledge effects making it possible to interleave planning with execution by invoking information-providing services during composition.

As the planning system relies on the inferences drawn by the DL reasoner, the practicality of the proposed solution crucially depends on the efficiency of the DL reasoner. For this reason, several novel optimization techniques, especially geared toward

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

handling nominals and large number of individuals, are presented. The other frequently used reasoning service by the HTN-DL planning system is conjunctive query answering. To improve query evaluation times, optimization techniques for conjunctive query answering inspired by the techniques used in relational databases are presented.

2.3. Syntactic-based WSC approaches

Currently there are two main approaches in the field of syntactic-based WSC [14]:

Web Service Orchestration: combines available WSs by adding a central coordinator (the orchestrator) that is responsible for invoking and combining the single subactivities. An orchestration also describes how other WSs are composed in order to achieve the required functionality of the WS.

Web Service Choreography, instead does not assume a central coordinator but rather defines complex tasks via the definition of the conversation that should be undertaken by each participant; the overall activity is then achieved as the composition of peer-to-peer interactions among the collaborating WSs. A Choreography also describes the external visible behavior of the WS. Choreography languages are still in an introductory phase of definition .WS-CDL [15] is example of this approach.

One of the most important orchestration languages namely BPEL4WS is defined as follows. Though this approach can be also considered as a workflow modeling language, the classification based on syntactic is preferred over workflow for this approach.

- *BPEL4WS*: This syntactic-based language was designed to enable the coordination and composition of a set of WSs. Also this language is based on WSDL [4], which is essentially an interface description language for WS providers. In fact, BPEL4WS is a merge between XLang and WSFL, but all of them are considered as a web service flow language [16]. WSC using BPEL4WS enables the definition of a new web service by composing a set of existing ones. The interface of the composite service is described as a collection of WSDL *PortTypes*.

A BPEL4WS process defines the roles involved in a composition as abstract processes. A buyer and a seller are examples of two roles. They are expressed using partner link definitions. We can have a role for each web service that is composed and does some activity. In order to integrate services, they are treated as partners that fill roles [17].

BPEL4WS depends directly on the WSDL of the service. A business process defines how to coordinate the interactions between a process instance and its partners. Thus, a BPEL4WS process provides one or more WSDL services. The BPEL4WS process is defined only in an abstract manner, allowing only references to service *portTypes* in the *partnerLink* [7]. Each partner is characterized by a partner link and a role name. In sum, business process is used to create an organizer that point to each service endpoint that will be actually executed.

2.4. Ontology-based WSC approaches

The Semantic Web [18] allows the representation and exchange of information in a meaningful way, facilitating automated processing of descriptions on the Web. Annotations on the Semantic Web express links between information resources on the Web and connect information resources to formal terminologies. These connective structures are called ontologies.

Ontologies are used as data models throughout these types of approaches, meaning that all resource descriptions and all data interchanged during service usage are based on ontologies. Ontologies are a widely accepted state-of-the-art knowledge representation, and have thus been identified as the central enabling technology for the Semantic Web. The extensive usage of ontologies allows semantically enhanced information processing and support for interoperability. In this section we consider two principal approaches, namely:

- *OWL-S* is an OWL service ontology for describing various aspects of Web services [19]. OWL-S has tried to adopt existing Semantic Web recommendations yet still maintain bindings to the world of Web services by linking OWL-S descriptions to existing WSDL descriptions [20]. In the following, we describe the four top-level concepts of the OWL-S ontology which are illustrated in Figure 1.

SERVICE: The SERVICE concept serves as an organizational point of reference for declaring WSs. Every WS is declared by creating a SERVICE instance. It links the remaining three elements of a WS through properties like PRESENTS, DESCRIBEDBY and SUPPORTS.

SERVICE PROFILE: declares what a SERVICE does in order to advertise and serves as a template for service requests at a high level, therefore enabling discovery and matchmaking. The profile includes nonfunctional aspects such as provider information and the quality rating of the service. The most essential information presented in the profile, however, is the specification of what functionality the service provides. Information transformation is represented by *inputs* and *outputs*; the change in the state of the real world caused

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

by the execution of the service is represented by *preconditions* and *effects*. Inputs and outputs refer to OWL classes describing the types of instances to be sent to the service and the respective responses to be expected. A feasible problem is that the semantics of these conditions is not covered by the (description logics) expressivity of the OWL-S ontology itself, but by reference to these languages. So, parties need to consent on the language for expressing conditions and also the notions of a "match" which is not addressed in the standard.

SERVICE MODEL: SERVICE could be described by a SERVICE MODEL which describes how a service works to enable invocation, enactment, composition, monitoring, and recovery. The service model views the interactions of the service as a process. OWL-S distinguishes between single processes and composite processes. But, a feasible problem is that the semantics of the workflow constructs is not expressible in the description logics underlying OWL, for which reason this semantics has been externally defined [21].

SERVICE GROUNDING: In order to map to the Web service world, an OWL service can *support* a grounding which maps the constructs of the PROCESS MODEL to detailed specifications of message formats, protocols, and others. In fact, SERVICE GROUNDING describes how to use a WS (i.e. how clients can actually invoke it).

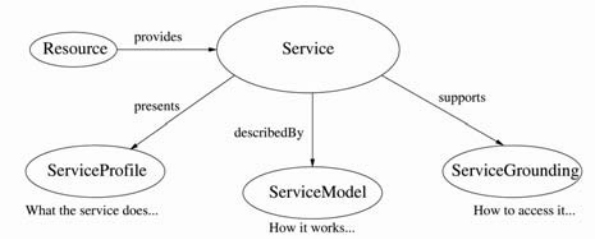

Figure 1. OWL-S conceptual model [19].

- *WSMO* defines a model to describe semantic WSs, based on the conceptual design set up in the WS Modeling Framework WSMF [22]. Following (Figure 2) the key aspects identified in the Web Service Modeling Framework, WSMO identifies four top-level elements as the main concepts [21]:

Ontologies: provide the (domain specific) terminologies used and is the key element for the success of Semantic Web services. Furthermore, they use formal semantics to connect machine and human terminologies.

Web services: are computational entities that provide some value in a certain domain. They are described from three different aspects: non-functional properties, functionality and behavior.

Goals: describe aspects related to user desires with respect to the requested functionality, i.e. they specify the objectives of a client when consulting a WS. Thus they are an individual top-level entity in WSMO.

Mediators: describe elements that handle interoperability problems between different elements, for example two different ontologies or services. Mediators can be used to resolve incompatibilities appearing between different terminologies (data level), to communicate between services (protocol level), and to combine Web services and goals (process level).

Besides these main elements, *Non-Functional* properties are used in the definition of WSMO elements that can be used by all its modeling elements. Furthermore, there is a formal language to describe ontologies and Semantic Web services called WSML (Web Service Modeling Language) which contain all aspects of Web service descriptions identified by WSMO. To introduce aspects of Semantic Web services in WSMO, the Meta-Object Facility (MOF) [23] specification is used, which defines an abstract language and framework for specifying, constructing, and managing technology-neutral metamodels. In addition, WSMX (Web Service Modeling eXecution environment) is the reference implementation of WSMO, which is an execution environment for business application integration. [24].

In the following, we describe the differences in the conceptual models of OWL-S and WSMO.

- OWL-S is specified using the Web Ontology Language, while WSMO uses an abstract MOF model. On the other hand, OWL-S defines its Meta model in the same language that it uses for concrete service descriptions. However, WSMO's basis in the abstract MOF model successfully avoids this problem.
- OWL-S does not separate what the user wants from what the WS provides. The service profile of a WS is not explicitly based on standard metadata specification. WSMO recommends the use of widely-accepted vocabularies (like the Dublin Core [25]).
- Non-functional properties in OWL-S are restricted to the service profile. However, this can be expressed in any WSMO element.

Figure 2. Four top-level elements of WSMO.

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

We give below some similarities in the conceptual models of OWL-S and WSMO:

- A service profile in OWL-S is close to a capability of a service or goal in WSMO. But, WSMO makes a conceptual distinction between the provider's and the requester's view, which is not made in OWL-S.
- The process model of OWL-S is conceptually similar to the WSMO service and goal interfaces. However, the distinction between the description of external behavior (choreography interface) and of the internal behavior (orchestration interface) is not made explicit in OWL-S.
- As for the grounding, WSMO and OWL-S adopt similar ideas with respect to binding to WSDL. However, the grounding is not a top-level concept in WSMO, but is instead integrated into the WSMO interfaces.

3. Comparative evaluation

In this section we compare the above WSC approaches with respect to the following criteria. We claim that, any approach to WSC should satisfy these set of criteria. The result can be seen in Table 1.

3.1. QoS

Currently, considering *quality of service* (QoS) to describe as nonfunctional properties is one of the critical issues in the WSC. When referring to QoS, nonfunctional properties such as performance, cost, or reliability are intended. Since a composed service uses other services to form itself, its quality depends on the WSs it uses. To be accepted by its customers, a business should try to provide good quality regarding the customers' requirements to a composed WS.

QoS aspects are considered when selecting WS candidates for a composition. By defining aggregation formulas for several QoS aspects which are applied to simple composition patterns, the whole workflow pattern of a composed service can be collapsed stepwise, and each time the most suitable collection of simple services is selected. As QoS information assigned with each basic service, *performance* and *availability* were chosen.

- *Performance:* This represents how fast a Web service request can be completed. According to [26], performance can be measured in terms of throughput, latency, execution time, and transaction time. The response time of a Web service can also be a measure of the performance. High-quality Web services should provide higher throughput, lower latency, lower execution time, faster transaction time and faster response time.
- *Availability:* the probability that a WS is available at any given time, measured as the percentage of time a WS is available over an extended period of time.

The management of QoS when composing WSs requires a careful consideration of the QoS criteria of the constituent WSs. To enable the specification and monitoring of QoS aspects like performance, financial, reliability, and availability, various approaches have been developed. An excellent research for considering QoS aspects in WSC can be found in [27]. Most of workflow based approaches like EFlow neglect specification of nonfunctional QoS properties such as *security*, *dependability*, or *performance*. In AI planning approaches such as Situation Calculus, a planning operator cannot represent such information. However, HTN-DL by using ontology that allows flexible matchmaking, tries to tackle this problem. Also, BPEL4WS does not directly support the specification of most QoS measures. However, in OWL-S, QoS measures such as availability are specified as service parameters in the WS description definition, but the specification of metrics and guarantees is missing. Moreover, there is no way to specify functional relations between metrics and therefore quality-aware WS discovery is not feasible [14].Finally, QoS (Nonfunctional properties) are applicable to all the definitions of WSMO elements such as *Ontologies*, *Web services*, *Goals*, and *Mediators*. Which QoS properties apply to which WSMO element is specified in the description of each WSMO elements.

3.2. Automatic composition

Many composition approaches aim to automate composition, which promises faster application development and safer reuse, and facilitates user interaction with complex service sets. With automated composition, the end user or application developer specifies a goal (a business goal expressed in a description language or mathematical notation) and an "intelligent" composition engine selects adequate services and offers the composition transparently to the user. The main problems are in how to identify candidate services, compose them, and verify how closely they match a request [28]. Generally, we cannot assign any of the above approaches as an automated approach. Although, most of these approaches like HTN-DL, OWL-S and WSMO can be assigned as a semi automated approach.

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

3.3. Composition scalability

This represents the ability of the WS to process multiple requests in a certain time interval. Composing two WSs is not the same as composing ten or more WSs. In a real-world scenario, end users will typically want to interact with many WSs while enterprise applications will invoke chains of possibly several hundred services. Thus, one of the important issues is how the proposed approaches scale with the number of WSs involved. It can be measured by the number of requests resolved in a certain time interval.

The HTN-DL, due to the fact that DL reasoner Pellet used, is optimized to handle large number of instances, and therefore has a tolerable scalability. In BPEL4WS, since XML files have increased a lot, WSC is a bit tiresome. BPEL4WS composition can be modularized, because this approach is recursive. But, BPEL4WS has no standard graphical notation. Some orchestration servers offer graphical representation for descriptions, such as UML, but they don't map one-to-one to complex constructs of BPEL4WS. Finally, OWL-S and WSMO have similar issues. The Web component approach achieves good scalability with class definitions, but requires additional time for mapping and synchronization between class definitions and XML.

3.4. Correctness

Verifying correctness depends on the WS and composition specifications. The composition of WSs may lead to large and complex systems of parallel executing WSs. An important aspect of such systems is the correctness of their behavior. Situation Calculus and HTN-DL, because of their solid mathematical basis in order to ensure the correctness of the compositions generated from the resulting planning domain, are very well. All other approaches offer no direct support for the verification of WSC at design time, to evaluate in this way its correctness. For example, BPEL is a Turing complete language dealing more with implementation than specification, and thus it's difficult to provide a formalism to verify the correctness of BPEL4WS flows [29]. The result is shown in Table 1.

Table 1. Comparing Web service composition approaches

	Criteria			
Approaches	QoS	(Semi) Automatic	Scalability	Correctness
EFlow	**Low**	**Low**	**Low**	**Low**
PPM	**Low**	**Low**	**Low**	**Low**
Situation Calculus	**Low**	**Low**	**Good**	**Average**
HTN-DL	**Average**	**Average**	**Good**	**Good**
BPEL4WS	**Average**	**Low**	**Average**	**Low**
OWL-S	**Good**	**Average**	**Good**	**Low**
WSMO	**Good**	**Average**	**Good**	**Low**

4. Conclusion

This paper has aimed to provide an overview and comparison for recent progress in WSC. We classify these approaches into four categories. But we cannot claim that this classification is exhaustive. In each category, the introduction and comparison of selected approaches are presented. The workflow-based approaches are usually used in the situation where the request has already defined the process model, but automatic program is required to find the atomic services to complete the requirement. The AI-planning based approaches deal with WSs as planning operators and use a causal planner to generate WSC. The syntactic-based approaches concentrate on two main approaches, namely: orchestration and choreography. Choreography languages are still in an introductory phase of definition. In ontology-based approaches, ontologies are used as data models throughout these types of approaches, meaning that all resource descriptions and all data interchanged during service usage are based on ontologies. The main problems with most of these approaches to compose Web services are the verification of correctness of WSC and the analysis of QoS aspects.

5. Acknowledgement

This research is supported by the Ministry of Science & Technology and Innovation (MOSTI), Malaysia and University of Technology Malaysia under the Vot. 79277.

6. References

[1] T. Berners-Lee, R. Fielding, and L. Masinter, *"*Uniform Resource Identifiers (URI): Generic Syntax*"*, *IETF RFC 2396.* [Online].Available: http://www.ietf.org/rfc/rfc2396.txt

[2] Web Services Architecture Requirements.(2004). [Online]. Available: http://www.w3.org/TR/wsa-reqs/

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

[3] New to SOA and Web services. [Online]. Available: http://www.ibm.com/developerworks/webservices/

[4] WSDL v1.1. (2001). [Online]. Available: http://www.w3.org/TR/wsdl

[5] SOAP v1.2. (2007). [Online]. Available: http://www.w3.org/TR/soap12-part1/

[6] F. Casati, M. Sayal, and M.C. Shan, "Developing E-Services For Composing Eservices", *In Proceedings of 13th International Conference On Advanced Information Systems Engineering (Caise)*, Springer Verlag, Interlaken, Switzerland, June 2001.

[7] T. Andrews et.al., BPEL v1.1.(2007).[Online].Available: http://www.ibm.com/developerworks/library/specification/ws-bpel/

[8] A. Ankolekar, DAML-S: "Web Service Description for the Semantic Web". *In Proceedings of ISWC'02*, ser. LNCS, vol. 2342, Springer Verlag, 2002, pp. 348–363.

[9] WSMO working group. [Online]. Available: http://www.wsmo.org

[10] J. Rao, "Semantic Web Service Composition via Logic-based Program Synthesis", *PhD Thesis*, Norwegian University of Science and Technology, Norway, 2004.

[11] J. Rao and X. SU, "A survey of automated web service composition methods". *In Proceedings of SWSWPC,* LNCS 3387, Springer, 2005, pp. 43-54.

[12] F. Casati, S. Ilnicki, and L. Jin, "Adaptive and dynamic service composition in EFlow". *In Proceedings of 12th International Conference on Advanced Information Systems Engineering (CAiSE)*, Springer Verlag, Stockholm, Sweden, June 2000.

[13] H. Schuster et.al, "Modeling and composing service-based and reference process-based multi-enterprise processes", *In Proceeding of 12th International Conference on Advanced Information Systems Engineering (CAiSE)*, Springer Verlag, Stockholm, Sweden, June 2000.

[14] M. Beek, A. Bucchiarone, and S. Gnesi, "Web Service Composition Approaches : From Industrial Standards to Formal Methods". *In Proceedings of Int'l Conf. On Internet and Web Application and Services (ICIW'07),* IEEE, 2007.

[15] N. Kavantzas, D. Burdett, and G. Ritzinger, WSCDL v1.0. (2004). [Online]. Available: http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/

[16] W.M.P. Van der Aalst, "Don't Go with the Flow: Web Services Composition Standards Exposed", *Intelligent Systems*, IEEE, 18(l), 2003, pp. 72-76.

[17] D.J. Mandel and S.A. Mcllraith, "Adapting BPEL4WS for the Semantic Web Bottom-up Approach to Web Services Interoperation", *In Second International Semantic Web Conference (ISWC)*, Sanibel Island, Florida, 2003.

[18] Berners-Lee, T., J. Hendler, and O. Lassila, *The Semantic Web*. ScientificAmerican, 2001, pp. 34–43.

[19] Fensel, D. et.al, *Enabling Semantic Web Services*, Springer, Berlin, 2007.

[20] *OWL-S: Semantic Markup for Web Services,* W3C Member Submission, (22 November 2004). [Online]. Available: http://www.w3.org/Submission/OWL-S/.

[21] S. Narayanan and S. McIlraith, "Simulation, verification and automated composition of web services", *In Proceedings of the 11th International World Wide Web Conference (WWW2002)*, Honolulu, Hawaii, 2002.

[22] Battle, S. *Semantic Web Services Framework (SWSF)*.

[23] Object Management Group Inc. (OMG), Meta Object Facility (MOF) Specification v1.4, 2002.

[24] WSMX working group. [Online]. Available: http://www.wsmx.org

[25] S. Weibel et al. (1998) Dublin Core Metadata for Resource Discovery. IETF 2413. [Online]. Available: http://www.ietf.org/rfc/rfc2413.txt

[26] Rajesh, S. and D. Arulazi, *Quality of service for Web services – demystification, limitations, and best practices.*

[27] D. ThiBen and P. Wesnarat, "Considering QoS Aspects in Web Service Composition", *In Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06)*, 2006.

[28] N. Milanovic and M. Malek, "Current solution for Web service composition", *IEEE*, 2004.

[29] X. Fu, T. Bultan, and J. Su, "Analysis of Interacting BPEL Web Services". *In Proceedings of WWW'04*. ACM Press, 2004, pp. 621–630.

[30] R. E. Fikes and N. J. Nilsson. "Strips: A new approach to the application of theorem proving to problem solving." Readings in Planning, Kaufmann, San Mateo, CA, 1990.

[31] S. Oh, D. Lee, and S. Kumara, "A Comparative Illustration of AI Planning-based Web Services

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

Composition", *In Proceedings of the 2005 ACM SIGecom Exchanges*, Vol. 5, No.5, December 2005.

[32] S. McIlraith and T. C. Son. "Adapting Golog for composition of Semantic Web services", *In Proceedings of the 8th International Conference on Knowledge Representation and Reasoning(KR2002)*, Toulouse, France, April 2002.

[33] F. Baader et.al, "Integrating description logics and action formalisms: First results", *In Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, Pittsburgh, PA, USA, 2005.

[34] E. Sirin. "Combining Description Logic Reasoning with AI Planning for Composition of Web Services", PhD thesis, University of Maryland, 2006.

978-1-4244-2328-6/08/$25.00 © 2008 IEEE